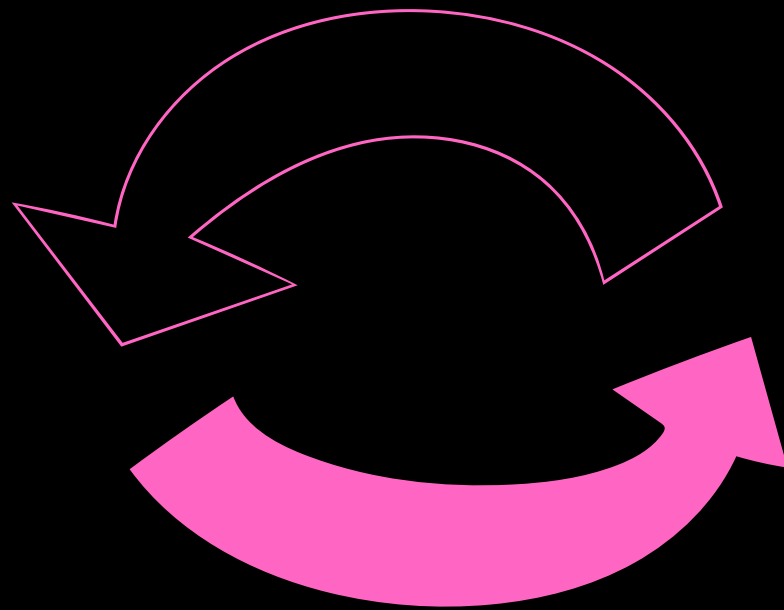
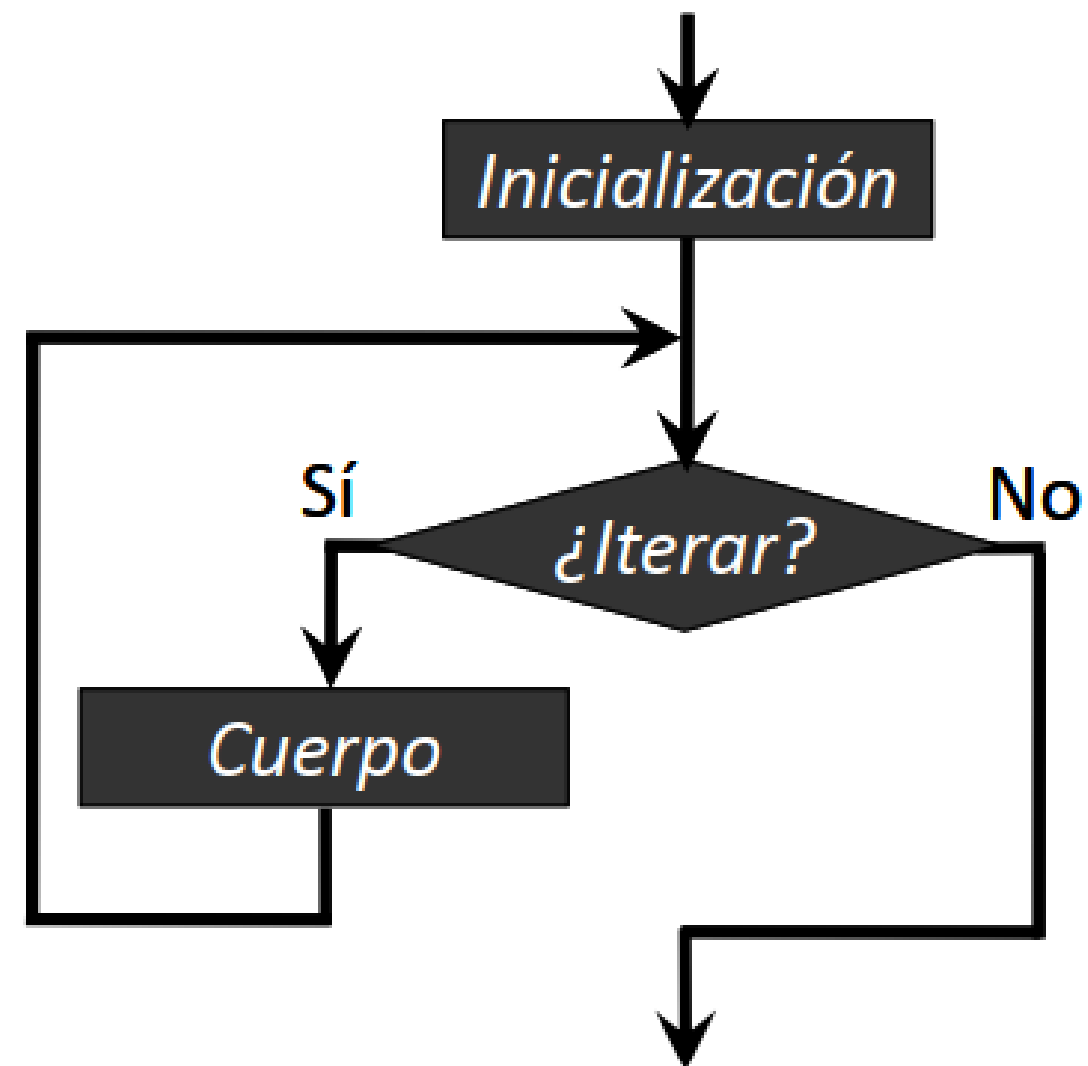
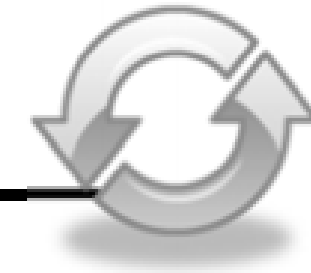


Instrucciones de repetición



Repetición

Repetición (iteración)



Bucles `while` y `for`

Tipos de bucles

✓ Número de iteraciones condicionado (*recorrido variable*):

- Bucle while

`while (condición) cuerpo`

Ejecuta el *cuerpo* mientras la *condición* sea true

- Bucle do-while

Comprueba la condición al final (lo veremos más adelante)

✓ Número de iteraciones prefijado (*recorrido fijo*):

- Bucle for

`for (inicialización; condición; paso) cuerpo`

Ejecuta el *cuerpo* mientras la *condición* sea true

Se usa una variable contadora entera

El bucle `while`

El bucle while

while.cpp

Mientras la condición sea cierta, ejecuta el cuerpo

```
while (condición) {
    cuerpo
}
```

Condición al principio del bucle

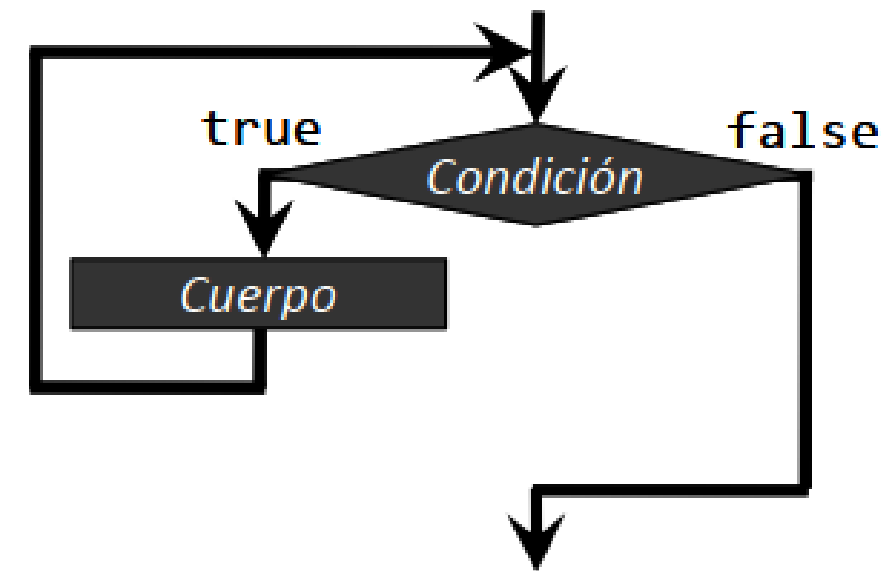
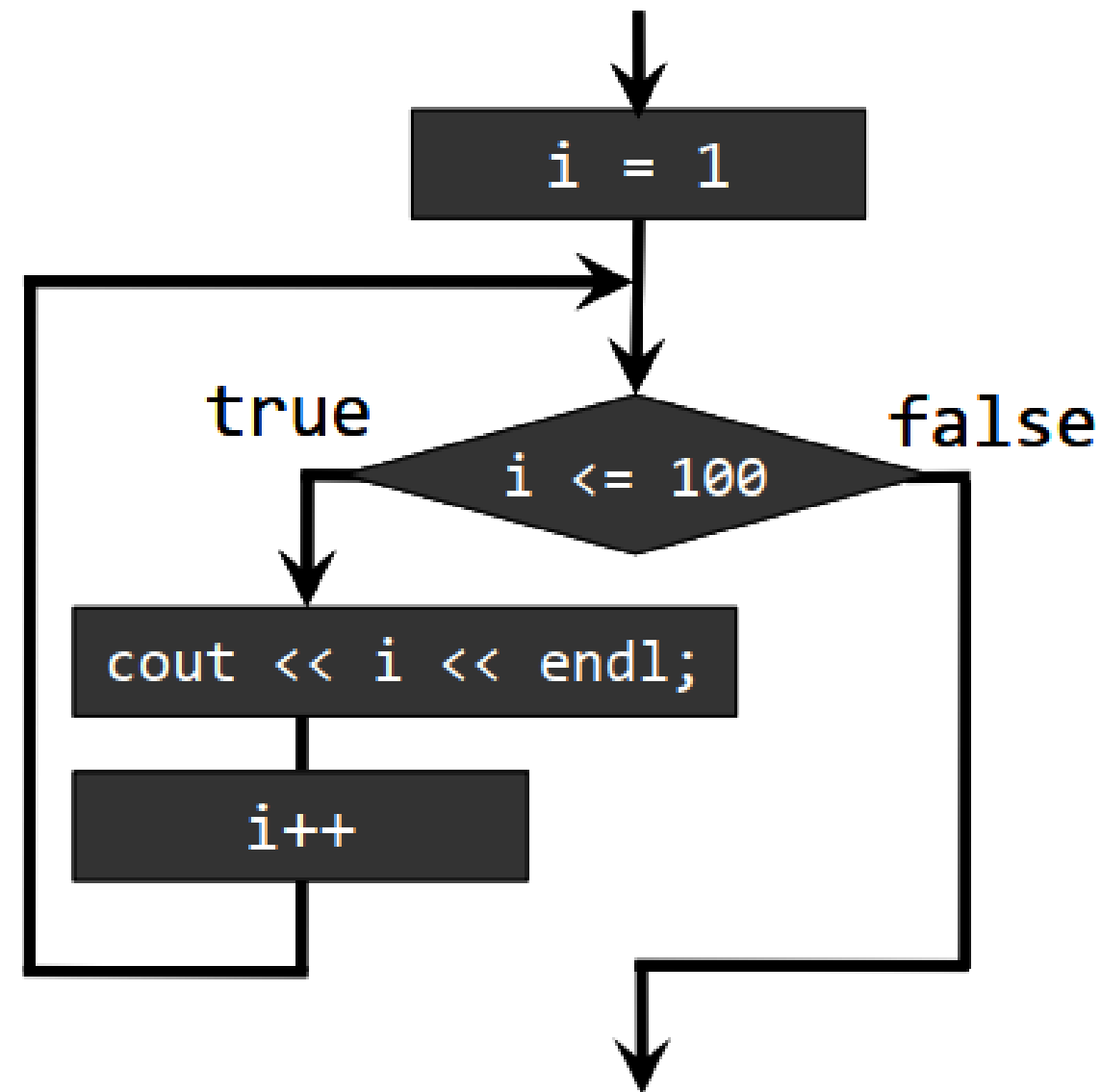
```
int i = 1; // Inicialización de la variable i
while (i <= 100) {
    cout << i << endl;
    i++;
}
```

Muestra los números del 1 al 100

Ejecución del bucle while

```
int i = 1;
while (i <= 100) {
    cout << i << endl;
    i++;
}
```

i 101



| |
|-----|
| 1 |
| 2 |
| 3 |
| 99 |
| 100 |
| — |

El bucle while

¿Y si la condición es falsa al comenzar?

No se ejecuta el cuerpo del bucle ninguna vez

```
int op;
cout << "Introduce la opción: ";
cin >> op;
while ((op < 0) || (op > 4)) {
    cout << "¡No válida! Inténtalo otra vez" << endl;
    cout << "Introduce la opción: ";
    cin >> op;
}
```

Si el usuario introduce un número entre 0 y 4:

No se ejecuta el cuerpo del bucle

Ejemplo de bucle while

primero.cpp

Primer entero cuyo cuadrado es mayor que 1.000

```
#include <iostream>
using namespace std;
```

*¡Ejecuta el programa para
saber cuál es ese número!*

```
int main() {
    int num = 1;
```

← Empezamos en 1

```
    while (num * num <= 1000) {
        num++;
    }
```

← Incrementamos en 1

```
    cout << "1er. entero con cuadrado mayor que 1.000: "
         << num << endl;
```

```
    return 0;
}
```

Recorre la *secuencia* de números 1, 2, 3, 4, 5, ...

Suma y media de números

sumamedia.cpp

```
#include <iostream>
using namespace std;
int main() {
    double num, suma = 0, media = 0;
    int cont = 0;
    cout << "Introduce un número (0 para terminar): ";
    cin >> num;
    while (num != 0) { // 0 para terminar
        suma = suma + num;
        cont++;
        cout << "Introduce un número (0 para terminar): ";
        cin >> num;
    }
    if (cont > 0) {
        media = suma / cont;
    }
    cout << "Suma = " << suma << endl;
    cout << "Media = " << media << endl;
    return 0;
}
```

Recorre la *secuencia*
de números introducidos

← Leemos el primero

← Leemos el siguiente

El bucle for

Bucle for

Número de iteraciones prefijado

Variable contadora que determina el número de iteraciones:

```
for ([int] var = ini; condición; paso) cuerpo
```

La *condición* compara el valor de *var* con un valor final

El *paso* incrementa o decrementa el valor de *var*

El valor de *var* debe ir aproximándose al valor final

```
for (int i = 1; i <= 100; i++)... 1, 2, 3, 4, 5, ..., 100
```

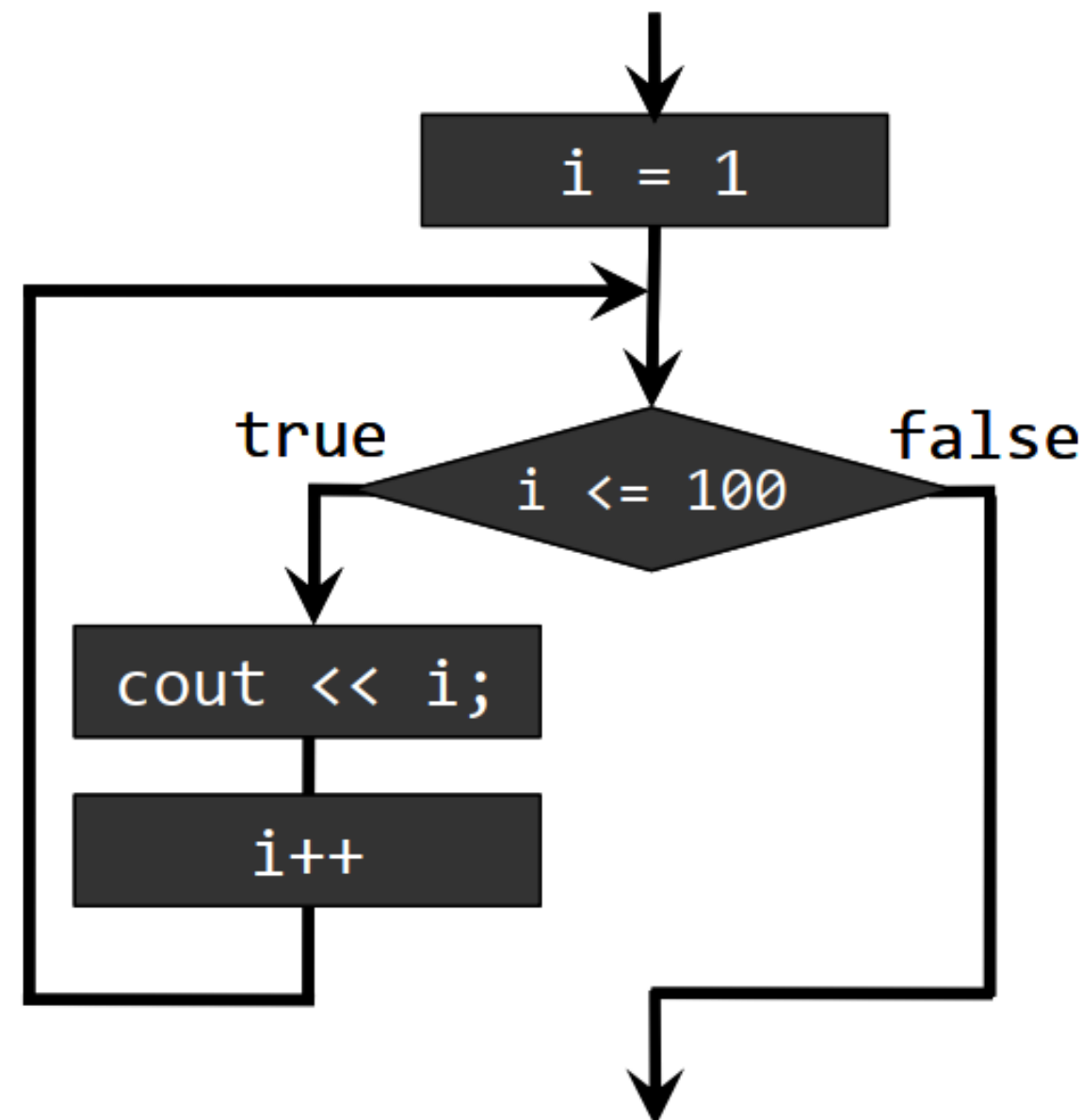
```
for (int i = 100; i >= 1; i--)... 100, 99, 98, 97, ..., 1
```

Tantos ciclos como valores toma la variable contadora

Ejecución del bucle for

for (inicialización; condición; paso) cuerpo

```
for (int i = 1; i <= 100; i++) {  
    cout << i;  
}
```



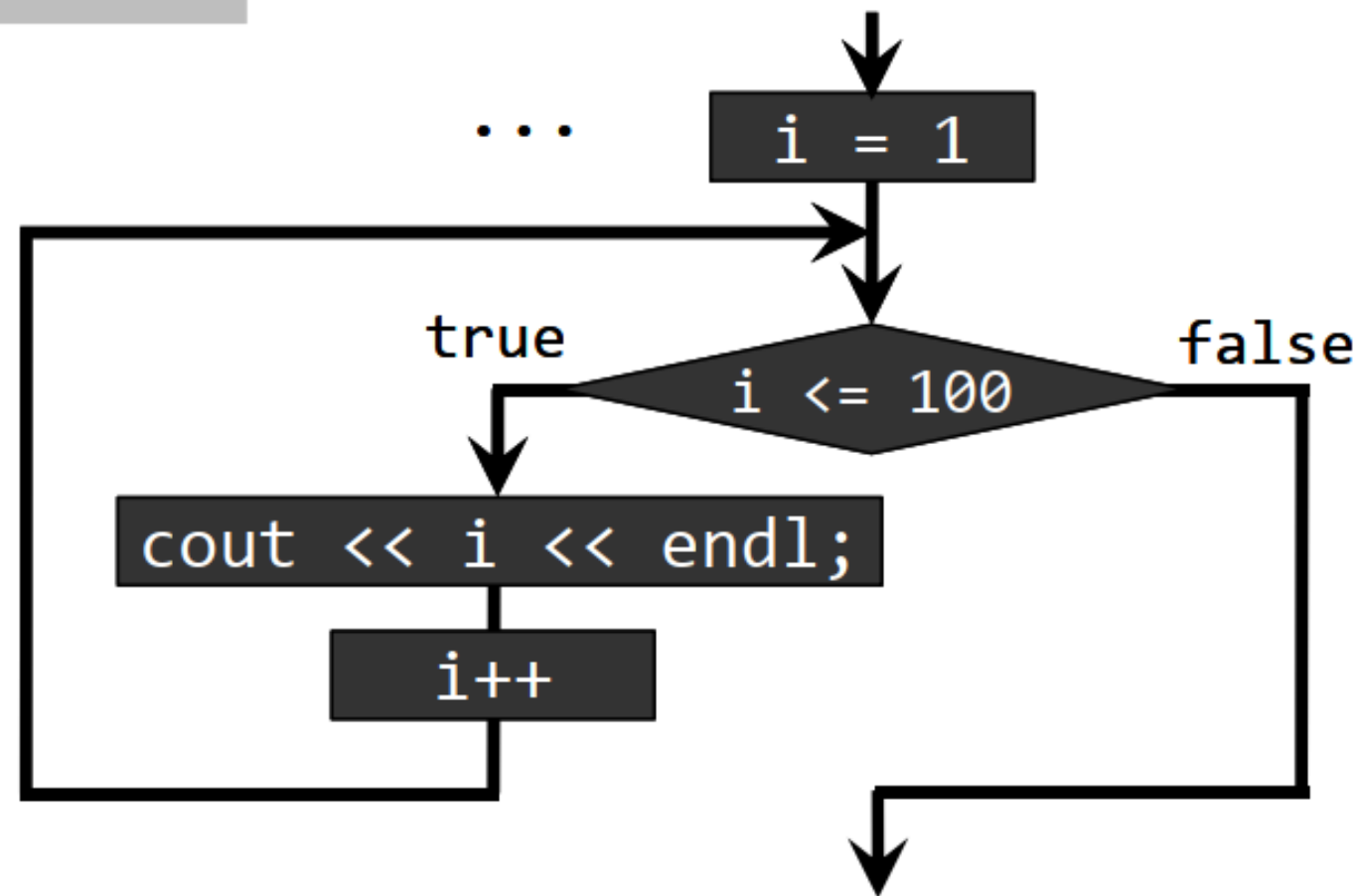
Ejecución del bucle for

for1.cpp

```
for (int i = 1; i <= 100; i++) {  
    cout << i << endl;  
}
```

i

101



| |
|-----|
| 1 |
| 2 |
| 3 |
| ... |
| 99 |
| 100 |
| — |

Bucle for

La variable contadora

for2.cpp

El *paso* no tiene porqué ir de uno en uno:

```
for (int i = 1; i <= 100; i = i + 2)
    cout << i << endl;
```

Este bucle for muestra los números impares de 1 a 99



Muy importante

El cuerpo del bucle **NUNCA** debe alterar el valor del contador

Garantía de terminación

Todo bucle debe terminar su ejecución

Bucles for: la variable contadora debe converger al valor final

Ejemplo de bucle for

suma.cpp

```
#include <iostream>
using namespace std;

long long int suma(int n);

int main() {
    int num;
    cout << "Número final: ";
    cin >> num;
    if (num > 0) { // El número debe ser positivo
        cout << "La suma de los números entre 1 y "
              << num << " es: " << suma(num);
    }
    return 0;
}

long long int suma(int n) {
    long long int total = 0;
    for (int i = 1; i <= n; i++) {
        total = total + i;
    }
    return total;
}
```

$$\sum_{i=1}^N i$$

Recorre la *secuencia* de números
1, 2, 3, 4, 5, ..., n

Bucle for

¿Incremento/decremento prefijo o postfijo?

Es indiferente

Estos dos bucles producen el mismo resultado:

```
for (int i = 1; i <= 100; i++) ...
```

```
for (int i = 1; i <= 100; ++i) ...
```

Bucles infinitos

```
for (int i = 1; i <= 100; i--) ...
```

1 0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -10 -11 ...

Cada vez más lejos del valor final (100)

Es un error de diseño/programación

Ámbito de la variable contadora

Declarada en el propio bucle

```
for (int i = 1; ...)
```

Sólo se conoce en el cuerpo del bucle (su ámbito)

No se puede usar en instrucciones que sigan al bucle

Declarada antes del bucle

```
int i;
```

```
for (i = 1; ...)
```

Se conoce en el cuerpo del bucle y después del mismo

Ámbito externo al bucle

Bucle *for* versus bucle *while*

Los bucles *for* se pueden reescribir como bucles condicionados

```
for (int i = 1; i <= 100; i++)  cuerpo
```

Es equivalente a:

```
int i = 1;
while (i <= 100) {
     cuerpo
    i++;
}
```

La inversa no es siempre posible:

```
int i;
cin >> i;
while (i != 0) {
     cuerpo
    cin >> i;
}
```

¿Bucle for equivalente?
¡No sabemos cuántos números
introducirá el usuario!

Bucles anidados

Bucles for anidados

Un bucle for en el cuerpo de otro bucle for

Cada uno con su propia variable contadora:

```
for (int i = 1; i <= 100; i++) {
    for (int j = 1; j <= 5; j++) {
        cuerpo
    }
}
```

Para cada valor de *i*
el valor de *j* varía entre 1 y 5

j varía más rápido que *i*

| <i>i</i> | <i>j</i> |
|----------|----------|
| 1 | 1 |
| 1 | 2 |
| 1 | 3 |
| 1 | 4 |
| 1 | 5 |
| 2 | 1 |
| 2 | 2 |
| 2 | 3 |
| 2 | 4 |
| 2 | 5 |
| 3 | 1 |

...

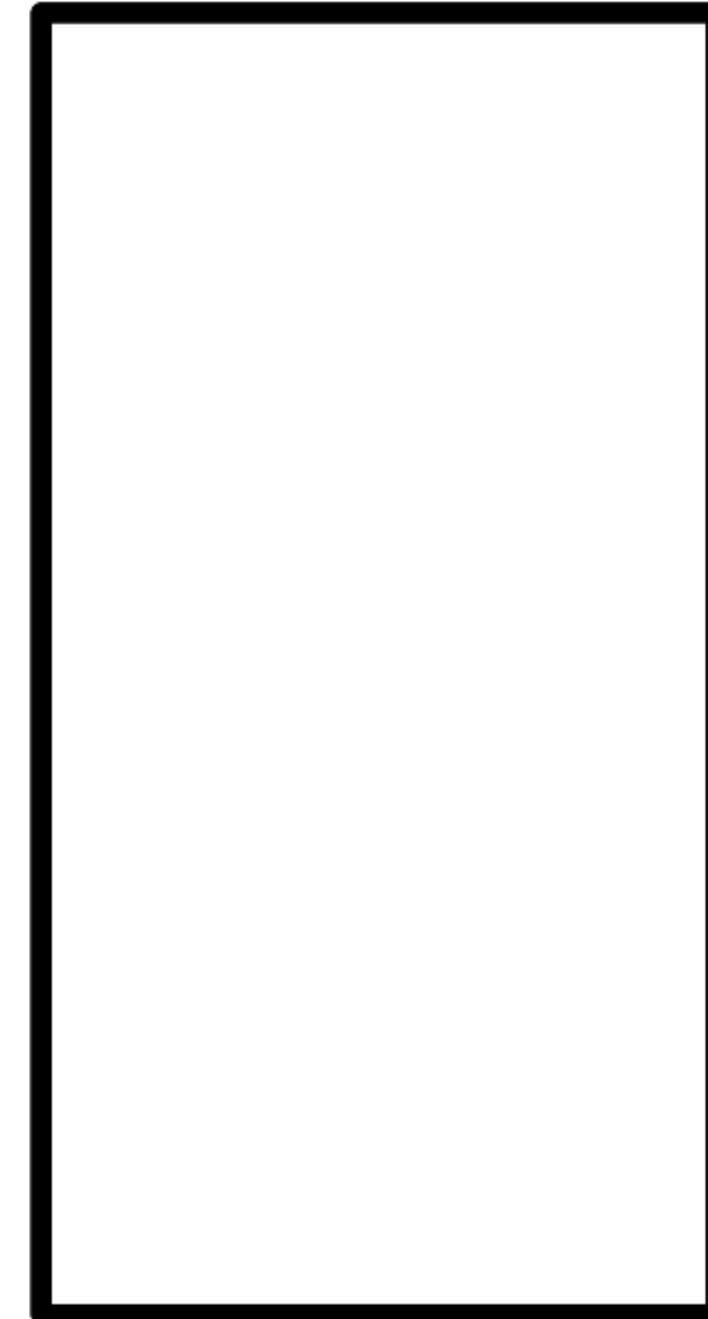
Tablas de multiplicación

tablas.cpp

```
#include <iostream>
using namespace std;
#include <iomanip>

int main() {
    for (int i = 1; i <= 10; i++) {
        for (int j = 1; j <= 10; j++) {
            cout << setw(2) << i << " x "
                << setw(2) << j << " = "
                << setw(3) << i * j << endl;
        }
    }

    return 0;
}
```



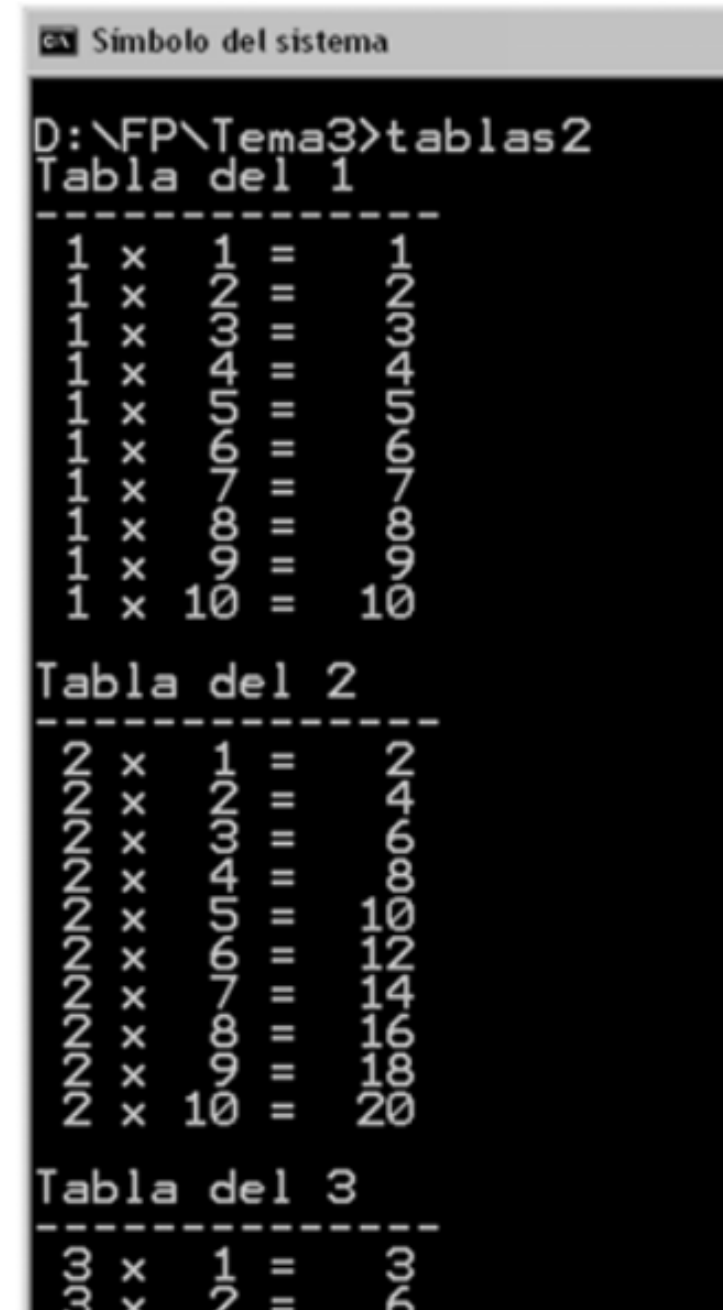
Mejor presentación

tablas2.cpp

```
#include <iostream>
using namespace std;
#include <iomanip>

int main() {
    for (int i = 1; i <= 10; i++) {
        cout << "Tabla del " << i << endl;
        cout << "-----" << endl;
        for (int j = 1; j <= 10; j++) {
            cout << setw(2) << i << " x "
                << setw(2) << j << " = "
                << setw(3) << i * j << endl;
        }
        cout << endl;
    }

    return 0;
}
```



```
GA Simbolo del sistema
D:\FP\Tema3>tablas2
Tabla del 1
-----
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10
Tabla del 2
-----
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
Tabla del 3
-----
3 x 1 = 3
3 x 2 = 6
```

```
#include <iostream>
using namespace std;
#include <iomanip>

int menu(); // 1: Tablas de multiplicación; 2: Sumatorio
long long int suma(int n); // Sumatorio

int main() {
    int opcion = menu();
    while (opcion != 0) {
        switch (opcion) {
            case 1:
                {
                    for (int i = 1; i <= 10; i++) {
                        for (int j = 1; j <= 10; j++) {
                            cout << setw(2) << i << " x "
                                << setw(2) << j << " = "
                                << setw(3) << i * j << endl;
                        }
                    }
                }
            break; ...
        }
    }
}
```


Más bucles anidados

```

    case 2:
    {
        int num = 0;
        while (num <= 0) {
            cout << "Hasta (positivo)? ";
            cin >> num;
        }
        cout << "La suma de los números del 1 al "
            << num << " es: " << suma(num) << endl;
    }
} // switch
opcion = menu();
} // while (opcion != 0)
return 0;
}

```

Más bucles anidados

```
int menu() {
    int op = -1;
    while ((op < 0) || (op > 2)) {
        cout << "1 - Tablas de multiplicar" << endl;
        cout << "2 - Sumatorio" << endl;
        cout << "0 - Salir" << endl;
        cout << "Opción: " << endl;
        cin >> op;
        if ((op < 0) || (op > 2)) {
            cout << "¡Opción no válida!" << endl;
        }
    }
    return op;
}

long long int suma(int n) {
    long long int total = 0;
    for (int i = 1; i <= n; i++) {
        total = total + i;
    }
    return total;
}
```

Ambos tipos de bucles anidados

```
while (opcion != 0) {  
    ...  
    for (int i = 1; i <= 10; i++) {  
        for (int j = 1; j <= 10; j++) {  
            ...  
        }  
    }  
    while (num <= 0) {  
        ...  
    }  
    for (int i = 1; i <= n; i++) {  
        ...  
    }  
    while ((op < 0) || (op > 2)) {  
        ...  
    }  
}
```

suma()

menu()

Ámbito y visibilidad

Ámbito de los identificadores

Cada bloque crea un nuevo ámbito:

```
int main() {
    double d = -1, suma = 0;
    int cont = 0;
    while (d != 0) {
        cin >> d;
        if (d != 0) {
            suma = suma + d;
            cont++;
        }
    }
    cout << "Suma = " << suma << endl;
    cout << "Media = " << suma / cont << endl;
    return 0;
}
```

3 ámbitos anidados

Ámbito de los identificadores

Un identificador se conoce
en el ámbito en el que está declarado
(a partir de su instrucción de declaración)
y en los subámbitos posteriores

Ámbito de los identificadores

```

int main() {
    double d;           Ámbito de la variable d
    if (...) {
        int cont = 0;
        for (int i = 0; i <= 10; i++) {
            ...
        }
    }
    char c;
    if (...) {
        double x;
        ...
    }
    return 0;
}

```

Ámbito de los identificadores

```
int main() {
    double d;
    if (...) {
        int cont = 0;    Ámbito de la variable cont
        for (int i = 0; i <= 10; i++) {
            ...
        }
    }
    char c;
    if (...) {
        double x;
        ...
    }
    return 0;
}
```


Ámbito de los identificadores

```
int main() {
    double d;
    if (...) {
        int cont = 0;
        for (int i = 0; i <= 10; i++) {
            ...
        }
    }
    char c;
    if (...) {
        double x;
        ...
    }
    return 0;
}
```

Ámbito de la variable i

Ámbito de los identificadores

```

int main() {
    double d;
    if (...) {
        int cont = 0;
        for (int i = 0; i <= 10; i++) {
            ...
        }
    }
    char c;
    if (...) {
        double x;
        ...
    }
    return 0;
}

```

Ámbito de la variable c

Ámbito de los identificadores

```

int main() {
    double d;
    if (...) {
        int cont = 0;
        for (int i = 0; i <= 10; i++) {
            ...
        }
    }
    char c;
    if (...) {
        double x;
        ...
    }
    return 0;
}

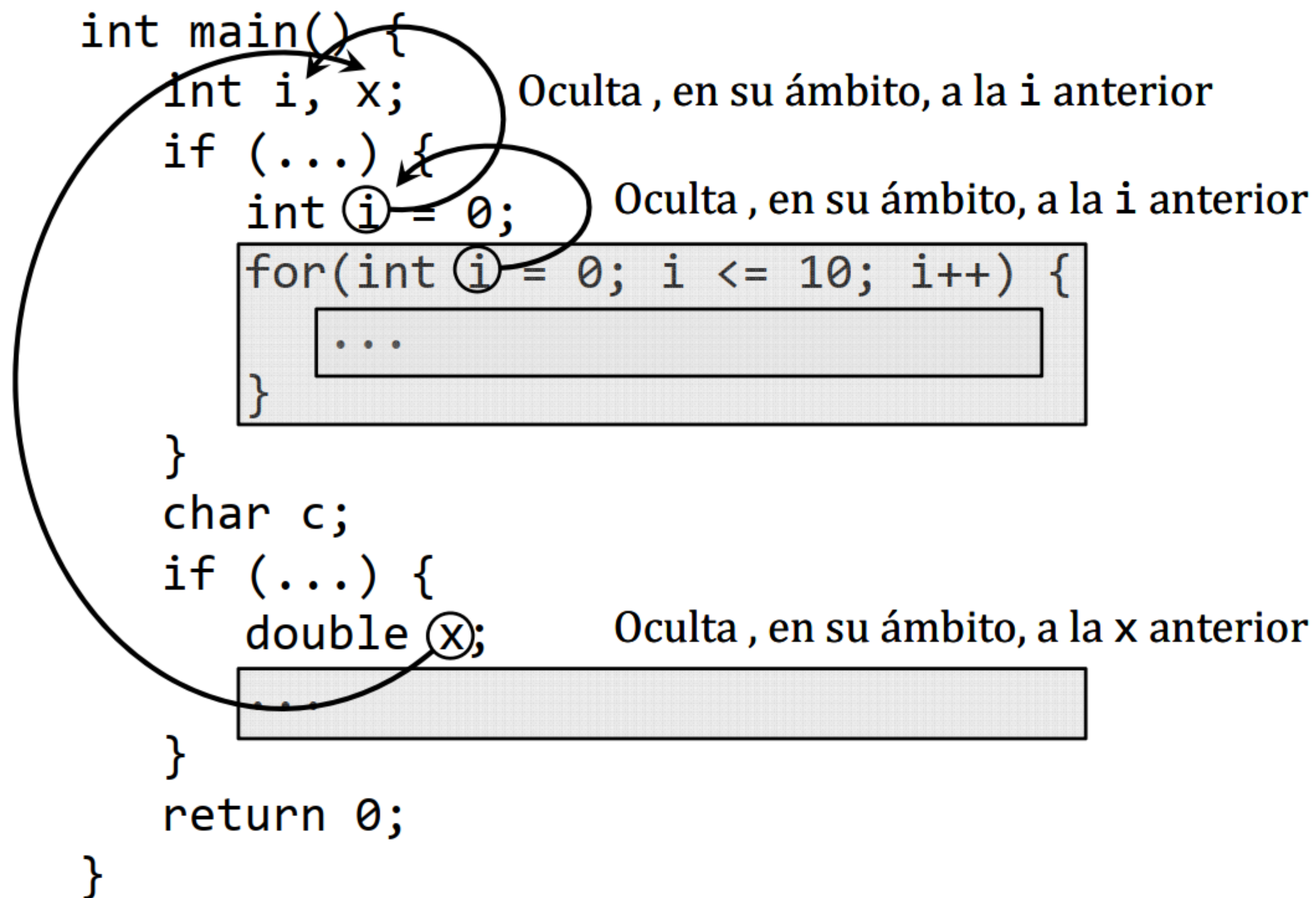
```

Ámbito de la variable x

Visibilidad de los identificadores

Si en un subámbito se declara
un identificador con idéntico nombre
que uno ya declarado en el ámbito,
el del subámbito *oculta* al del ámbito
(no es visible)

Visibilidad de los identificadores

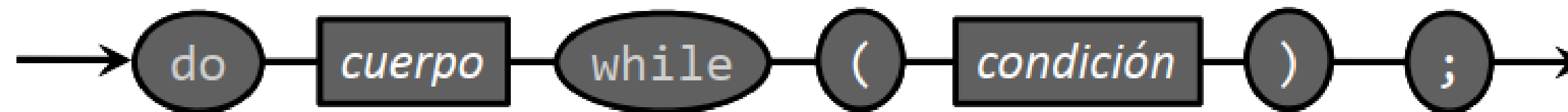


do-while

Otro bucle no determinado de C++

El bucle do..while

do cuerpo while (condición); Condición al final del bucle



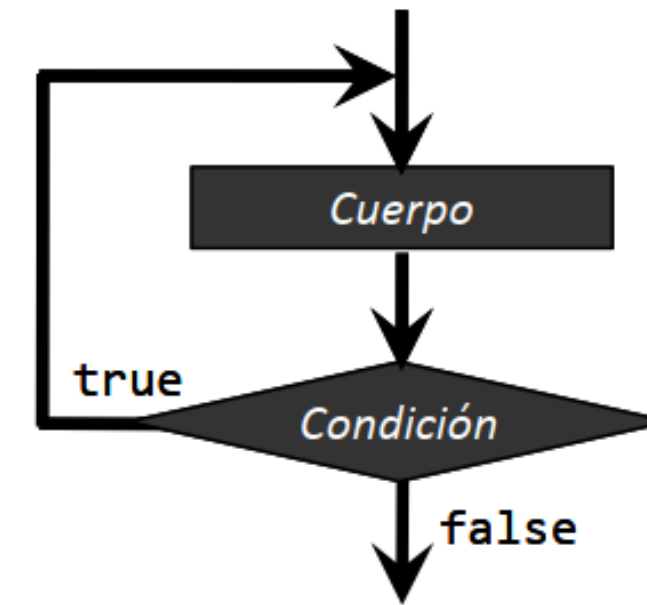
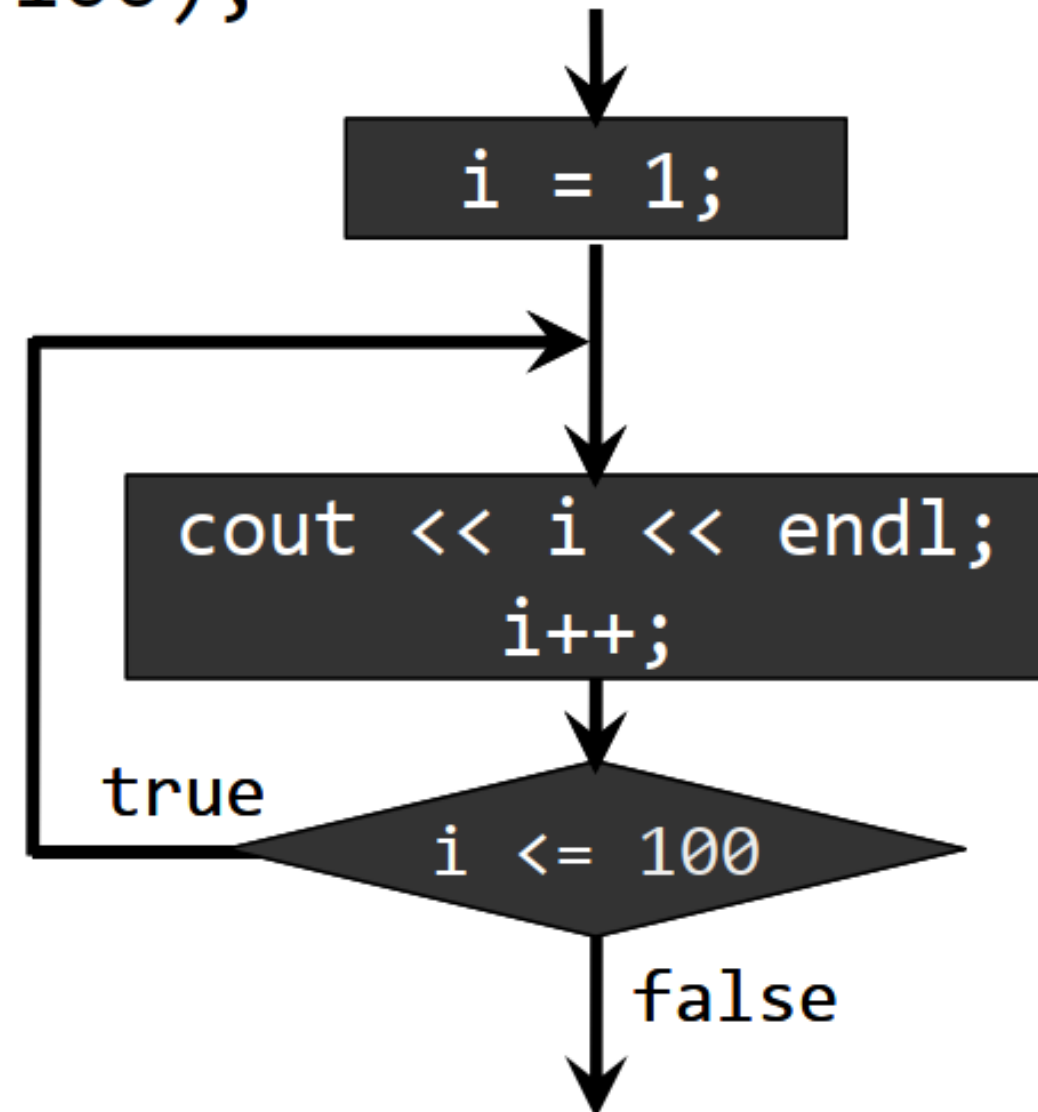
```
int i = 1;
do {
    cout << i << endl;
    i++;
} while (i <= 100);
```

El *cuerpo* siempre se ejecuta al menos una vez

El *cuerpo* es un bloque de código

Ejecución del bucle do-while

```
int i = 1;
do {
    cout << i << endl;
    i++;
} while (i <= 100);
```



El cuerpo
se ejecuta
al menos
una vez

while versus do-while

¿Ha de ejecutarse al menos una vez el cuerpo del bucle?

```
cin >> d; // Lectura del 1º
while (d != 0) {
    suma = suma + d;
    cont++;
    cin >> d;
}
```

```
do {
    cin >> d;
    if (d != 0) { // ¿Final?
        suma = suma + d;
        cont++;
    }
} while (d != 0);
```

```
cout << "Opción: ";
cin >> op; // Lectura del 1º
while ((op < 0) || (op > 4)) {
    cout << "Opción: ";
    cin >> op;
}
```

```
do { // Más simple
    cout << "Opción: ";
    cin >> op;
} while ((op < 0) || (op > 4));
```

El menú de la aplicación con do-while

```
int menu() {
    int op;

    do {
        cout << "1 - Añadir un nuevo estudiante" << endl;
        cout << "2 - Eliminar un estudiante" << endl;
        cout << "3 - Calificar a los estudiantes" << endl;
        cout << "4 - Listado de estudiantes" << endl;
        cout << "0 - Salir" << endl;
        cout << "Opción: ";
        cin >> op;
    } while ((op < 0) || (op > 4));

    return op;
}
```